DOCUMENT RESUME

ED 060 616 EM 009 622

AUTHOR Bork, Alfred M.; Ballard, Richard
TITLE Computer Graphics and Physics Teaching.
INSTITUTION California Univ., Irvine. Physics Computer Development Project.
SPONS AGENCY National Science Foundation, Washington, D.C.
PUB DATE 25 Feb 72
NOTE 9p.

EDRS PRICE MF-$0.65 HC-$3.29
DESCRIPTORS *Computer Assisted Instruction; *Computer Graphics; Computer Programs; Display Systems; Instructional Innovation; *Physics Instruction; Program Descriptions; Programed Instruction

ABSTRACT

New, more versatile and inexpensive terminals will make computer graphics more feasible in science instruction than before. This paper describes the use of graphics in physics teaching at the University of California at Irvine. Commands and software are detailed in established programs, which include a lunar landing simulation and a program which teaches the laws of motion. Graphic teaching is held to be more intuitive than nongraphic and the possibility of student-written graphics (once software is perfected) is considered favorably. (RB)

COMPUTER GRAPHICS AND PHYSICS TEACHING

Alfred M. Bork and Richard Ballard
Physics Computer Development Project
University of California, Irvine
Irvine, California, 92664
Phone: 714-833-6911

January 11, 1972; Revised February 25, 1972

Last year at the Dartmouth Conference[1] we reported on the Physics Computer Development Project's use of computers in learning physics. We are concerned with procedures and areas in which the computer gives leverage in teaching difficult to obtain any other way. Our work has encompassed all modes of computer usage: dialog, computation, and simulation. Furthermore we have worked with full classes, testing our material with 150 students and rewriting it based on computer-saved feedback.

Our early work was with character-oriented terminals, either hardcopy or softcopy.[2] However, from the beginning we were interested in using graphic terminals; and recently our software development has permitted us to implement and use graphic teaching material for physics courses. This paper describes the types of usage of graphics in teaching we are exploring and the underlying software for graphics.

Pictures in Teaching

By looking in any textbook or visiting any lecture we can see that pictures, diagrams, and graphs are useful in teaching. The crude ability of alphanumeric terminals to present graphics has often been invoked in educational situations. Even a terminal such as the Model 33 Teletype can simulate point graphs by typing characters in fixed positions. Some of our dialogs produced crude diagrams and students often tried to graph output from their own programs.

Graphics in a timesharing environment has been expensive. However, new terminals and spectacular decreases in price promise that graphics will soon be widely available for students. These terminals can, under computer command, draw a line from one point to another on the screen and thus draw pictures of arbitrary complexity. Little use has been made of graphics with live students, so little is known about its effectiveness in teaching environments.

It is convenient to distinguish two types of computer usage in teaching, one in which the student writes his own programs and another in which the student interacts with existing programs. Implementation of graphic facilities is different in these two cases, and perhaps they even have different educational values.

Most of the work described here deals with this last type of usage. We will look first at the underlying graphic software used in dialog programs, then at comparisons of popular dialog programs which exist in both graphic and nongraphic forms. We next look at dialogs in which graphics is the key element. MOTION, a program for exploring classical mechanics, is given special attention. At the end of this paper we discuss briefly an interesting possibility for graphics within student written programs. As of this writing, the graphic software for this use is in early stages of implementation.

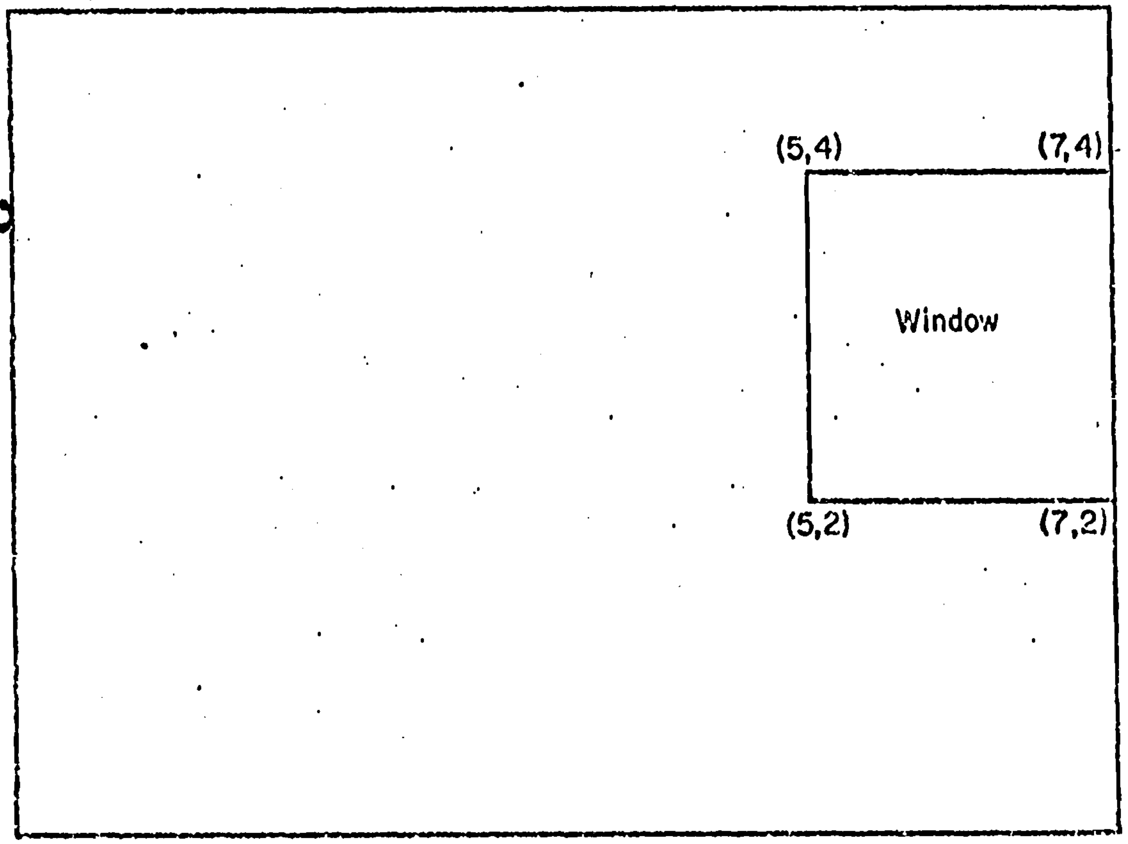
Underlying Graphic Software for Dialogs

Our development of graphic software was an extension, within the same tradition, of our software for generating student-computer dialogs without graphics. Our approach has been to write assembly language macros which access assembly language subroutines. As teaching needs develop, we write new macros; so graphic teaching is a special case for our general tactic.[3] We make no attempt to give a complete software description; full documentation exists for those interested.[4] We illustrate some of the principal graphic macros, available under the BTM and UTS timesharing systems on the Xerox Sigma 7.

aphic data can be computed within the program or fixed drawings
) be part of the program. Typically in our programs the data is computed in code which originated as FORTRAN subroutines. The graphic data from the subroutine is in arrays.

The teacher might first decide where the picture is to appear on the screen. We might want to put several curves on at one time and to mix graphic with alphanumeric material. So the teacher needs an easy way of controlling where things appear. (In practice this is complicated by the fact that screens have different sizes and orientations; although our software covers this, we will not discuss it).

We let the user specify where he wants the curve drawn by a WINDOW command with specifications in inches from the lower left hand corner. Suppose, for example, that he wants a curve, perhaps one of several, to appear in the box or window illustrated in the following diagram:



The statement within the program to establish this window would be:

```
WINDOW    (5,2),(7,4)
```

The user can also specify a box around the window if he so desires:

```
WINDOW    (5,2),(7,4),BOX
```

Normally the box will not be drawn.

The teacher must next decide <u>where</u> within the window the curve is to be. Possibilities are numerous: we can choose to have the x and y or x, y and z data scaled so that it occupies the full window. Or the origin of the coordinate system can appear at the center of the window. Or the teacher may specify the coordinates of the ends of the window, again in two or three dimensions. The following uses of the macro SCALE let the user assign coordinates to the window:

```
SCALE     (X1,X2),(Y1,Y2)

SCALE     (P,Q),(X,Y),(A,B)
```

P & Q are the minimum and maximum for the first variable plotted, X & Y for the second, and A & B for the third.

After establishing the window and the scale, we next draw the curve. The data will be in two or three arrays; the FORTRAN routine also returns us the number of points to be plotted. The command CURVE connects each of the "points" in the arrays with straight line segments. If the curve is three dimensional, it is projected onto the two dimensional screen.

Typical uses of the CURVE command are as follows:

CURVE	(X,Y,N)	2-D plot
CURVE	(HOR,VER,OUT,N)	3-D plot
CURVE	(MAX,(X,Y,N))	3-D plot, curve is scaled to touch window on all sides.
CURVE	(CENTER,(AA,BB,CC,M))	(o,o) is centered in the window.

The fourth command, the last we describe in detail, is AXES, drawing axes for two or three dimensional curves. Here are some examples:

AXES		2-D axes using current scaling data
AXES	(DIM,3)	3-D axes
AXES	(MAX,A,B,50)),LIMITS Axes for largest possible curve. Maximum and minimum values of axes are shown.	
AXES	(LABELS,'X','Y')	
AXES	(LABELS,'VX','VY','VE')	

Other macros are needed for positioning the beam, for erasure, and for specifying the graphic terminal; current software supports the ARDS 100, the Tektronix 4002, the Tektronix 4002A, and the Tektronix 4010. Adding a new terminal is a simple modification. A graphic program starts by asking the student which type of terminal he is using; with present terminals, unfortunately, the computer has no way of knowing the nature of the terminal. (We have information available,[5] for those interested in the problem, about terminals for an educational environment.)

Graphic and Nongraphic Dialogs

Several dialogs which use graphics are available in both graphic and nongraphic form; this gives us the opportunity to comment on the effectiveness of graphics. If a program exists in both forms, we would not expect it to be the same program, because the availability of new facilities indicates different possibilities in the teaching environment. Particularly in the early period of understanding the use and effect of graphics in learning it is valuable to have some similar programs attempting to exploit both graphic and nongraphic environments.

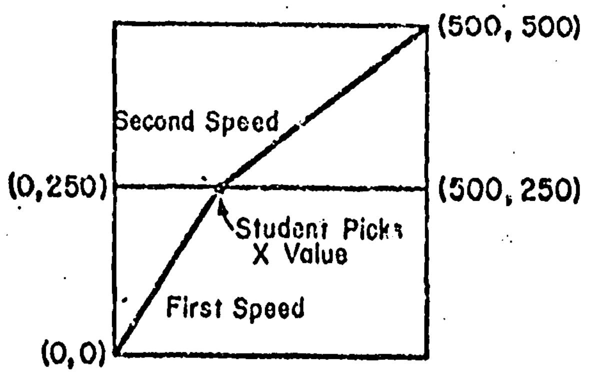
One dialog in both forms is the widely available one-dimensional lunar landing simulation. We have a nongraphic version, with numbers coming out, plotting the position in the usual typewriter way. This was written for our system by Noah Sherman and Steven Derenzo at the Lawrence Hall of Science, University of California, Berkeley.

The graphics lunar landing simulates a stylized spacecraft panel, with the position, velocity, and fuel indicated on graphs or gauges. The "instruments" are changed when the pilot gets close to the lunar surface, so that the student can get a more detailed view.

Both versions are popular with students; this is our most widely used program, both with physics students and others. It is clear from observing students that physics talent needed in one program differs from that needed in the other. In the "numbers" program, without graphic output, good students make "$1/2\ at^2$" calculations to determine fuel requirements in the final stages of landing; it was in connection with getting experience with the relations involving motion at constant acceleration that the dialog was developed. The graphics dialog makes students rely on curves of position and velocity vs. time and so has an entirely different "feel" to it. Students no longer make calculations but must develop an intuitive idea of what it is like to be involved in a constant acceleration environment with some fuel to slow down. We suspect, although we could not prove, that students learn more in the graphic

vironment in developing an intuitive feeling for the laws of motion. We contemplate tests involving these two versions.

A second dialog available in both graphic and non-graphic forms was developed by Murray Alexander of De Anza College in Cupertino, California. It is a "race," the Fermatopolis 500. In this somewhat unusual race the drivers have no control over the speed in the two laps. They go from (0,0) to (500,500) as shown in the following diagram:



A change in speed occurs only at $y = 250$. The speeds are announced in advance for each race for each of the two areas, and are the same for both "drivers." Each driver picks x corresponding to $y = 250$. The object is to win the race by keeping the time of travel as short as possible. The physicist sees that a minimal principle is involved, Fermat's principle in optics and leading to Hamilton's principle in mechanics. Variational principles are certainly an important way to formulate physical laws. Here is an important physical idea, vital in contemporary physics, almost totally neglected in the vast majority of introductory courses. Hence the computer has an opportunity to contribute significantly to learning in physics. Even a professional quickly discovers he is better off using his intuition than attempting to make the

calculation, and the nonprofessional quickly sees what is involved in finding a minimum time in such a situation. The reward system has a higher payoff if the student's time is closer to the minimum; winning or losing depends on several races, with different speeds in the two regions.

With FERM it is more difficult to be definite about the advantages and disadvantages of the graphic versus the nongraphic form. In the graphic form you <u>see</u> the race happening, while in the nongraphic version you see only the resulting times; we believe that you get some feel as to <u>why</u> you win in the graphic version, because you observe that the winning person travels farther in the faster region. So there seems to be a gain in the graphics, but perhaps this gain is not sizable. Again we intend to do some testing with students using both versions, to see which form more quickly develops the students' intuitive understanding of action principles. We hope too to develop increasingly complex follow-up games of the same type, further extending the notion of variational principles.

GRAPH performs a utility function for students, one that we feel is very valuable for physics classes. As its name indicates, it graphs functions. Students <u>should</u> do some graphing by hand, but it is difficult to generate large numbers of graphs by hand. Yet seeing relations graphically is often an important part of understanding the physical aspects of a mathematical result. GRAPH makes it possible to examine many curves concurrently. Students enter the function in the usual notation with relatively few restrictions. The program contains its own parser which analyzes the functions and generates the graphic data. Students can request many different plots, set constants in equations to different values, etc. Functions are in parametric form, and plotting in two and three dimensions is available as in all of our graphic material.

This program has also found a use entirely different from that initially intended, in a "physics for artists" class. If we connect points on a curve which are not close together, we can construct beautiful patterns. The program allows students to control all variables, including the time step between successive points to be plotted.

MOTION

MOTION is a more elaborate instructional program using the computation, language recognition, and graphic facilities of the computer. In combination they have produced an exciting new tool and introduced several new teaching strategies.

MOTION was written to aid students and instructors at all levels in a study of equations of motion. The program offers each user a large repertoire of motions. Choices range from simple harmonic, central force, and constant acceleration to the very uncommon motions associated with two force centers or a multipole field. One can view the effects of anharmonicity, uniform electric and magnetic fields, or even the scattering from a nuclear force. A revised version will use parsing routines permitting students to write their own equations of motion.

Classical mechanics offers unique opportunities for using the computer to carry us far beyond present course boundaries. We have had ample demonstrations here and elsewhere that simple numerical methods, like the Euler method, can be taught to students at every level.[8] It is difficult to overstate the potential significance of introducing students so early to so powerful a tool. The beginning student can tackle problems and physical systems heretofore reserved to graduate students.

We are only just beginning to exploit the opportunities that a numerical approach provide. We have developed considerable experience in using these methods with large classes of students.[1] We have probed many of the pedagogical barriers to a wider use of computers. MOTION was written to overcome what seemed the more serious of these problems.

Perhaps the dominant objection to numerical solutions is that they produce a numerical result. For the most part it is difficult to interpret such results--to extract their physical consequences and go on to an [illegible]pate the solutions of other problems.

Wheeler proclaims as his First Moral Principle,[9] "Never make a calculation until you know the answer." For many of our students, development of a physical intuition and an appreciation for the range of physical phenomena will serve them better than a knowledge of the mechanics for producing a particular solution.

With numerical solutions all results appear the same on the teletype, simple columns of numbers. Students rightly balk when asked to translate these numbers into graphs. They often view the process of changing parameters and replotting as tiresome busy work, yet repetitous plotting is the key element in learning from numerical solutions.

Using MOTION

MOTION uses a graphic dialog approach to overcome this objection. Students acquainted with the underlying algorithm can use that knowledge to explore the sensitivity of solutions to time step and other considerations. Other users, knowing nothing of such matters, will never encounter them. Using simple English they can choose motions for study, change constants and initial conditions, and then observe consequences of such changes.

The program can provide a form of instant experience to the student. In a very short time he can develop a qualitative understanding, for example, of any central force described by a power law or the sometimes spectacular orbits of a planet in a binary star system. The student is not restricted to plotting only spatial variables, nor in the choice of two and three dimensional projections. Virtually any physically meaningful variables can be plotted against any one or two other variables.

The explorative characteristics of an instructional program like MOTION are very important. Students bring to their physics classes a very narrow range of experience. While this has always been true, it has become increasingly acute as physics moves on to microscopic and macroscopic levels far removed from everyday

observation. Students need this experience to understand physical principles. Great laws only appear as such when they help us to consolidate a variety of seemingly unrelated observations. MOTION offers a rich universe of examples. The unique behavior of total energy is nowhere more impressive than in three body motion. Its straight line time dependence stands out strikingly against the bizzare trajectories traced by other variables. We provide a wide range of physical examples, some obvious in their conservation of energy and in momentum, some not.

The sense of exploration in MOTION is quite real. If one ignores the infinity of variations produced by changing initial conditions and equation constants, over a hundred and fifty thousand distinctly different combinations of equations and variable projections are possible. Most of these have never been seen before by anyone. As a consequence, every user has the opportunity to learn something new and make genuine discoveries. Unlike most instructional programs, both the instructor and the student users are offered an opportunity to learn. If anything, the instructor's knowledge and experience may permit him to learn even more than the student. This last aspect has been exceedingly important in gaining faculty acceptance for the program. Instructors can test the effectiveness of the program on themselves.

Response Recognition in MOTION

Explorative programs like MOTION are difficult to program. The bulk of the computation and display options are straightforward; the tricky question is how to educate the user in the existence and operation of so vast a collection of options: all the equations, variables, projections, scaling, families, 3-D aids--like rotation and dashing. We chose a dialog strategy that puts the student in control of the program flow, letting him call for the facilities he wants. We take full advantage of dialog technique as a means of producing stand alone programs. It requires no prior instruction or descriptive handouts and adapts to the terminlogy and abilities of an enormous range of users.

MOTION attempts to recognize any question or request relating to its functions of selecting, solving, displaying motion. Although this sounds to be most difficult, it is not an impossible task. It departs completely from the patterns of program flow found in computer dialogs on programmed instruction. Students accustomed to these conventional dialogs will sometimes ask, "Where am I in the program? What can I do next?" The answer is that despite appearances they are always at the same point and that they are free to try anything they want.

Each input is inserted into a ring which performs an exaustive search for key word fragments or symbols. As successive requests or questions are often related, the search process is made most efficient by inserting the input adjacent to the last successful key match. Suppose the student's last input was recognized as assigning a new value to one of the initial conditions. He is more likely to change another or to ask for a "plot" than to request another equation of motion. The program will do either, but checks first for the most logically related.

Once the presence of one or more key word fragments or symbols has been detected, subroutines are called to break down the message syntax. If parts are missing, the program requests their entry. Here again we try to avoid any "flow traps." We look first for the missing information; if not present, we reinsert the new input into the test ring, on the chance that the user has disregarded our question and changed the subject.

These recognition facilities have proven quite effective. Anyone who knows what he wants, can ask for it. If clearly stated, he will usually get it. He need change only those things he wishes changed; all others will remain the same. If he does not set equation constants or initial conditions before asking for a plot, the program loads in an interesting example and proceeds to plot it.

MOTION employs several techniques for enlarging the student's knowledge of its facilities. The imaginitive user will ask questions. This does not come easily; many users put off "wasting their time" until a high level of responsiveness has been demonstrated. Some reject the notion entirely. This may be the result of a previous exposure to computers; the totally uninitiated often accept the idea with great glee and begin asking questions having little relationship to the program.

Questions are usually answered with an excess of information. The requested facility is described and notice is taken of other related facilities perhaps unknown to the user.

Students having trouble with the program are also given an opportunity to learn. Whenever an input cannot be recognized, failing all tests; the program randomly selects a message appropriate to that area of the program. It describes some of the available features, using quotes to emphasize recognized terminology.

Observations on Student Use of MOTION

MOTION has been used by a spectrum of students, instructors, and computer professionals. It is, first of all, highly popular. Left to themselves, many students have spent the better part of a day running the program and return frequently thereafter. It was immediately adopted, highly recommended, and heavily used in the upper division mechanics course. Previously, the instructor had seen little use for computers in teaching. We are now testing the program with large classes at the introductory level.

The ways of using MOTION are varied. Instructors could connect the graphic terminal to a scan-converter and use the program as a televised demonstration in lectures. The format free, natural language approach in communications makes it very easy for instructors to learn its use; the absence of flow keeps them from wasting class time, if an error is made.

It is most often used by students as an adjunctive aid, available at any time. Students move easily between physical systems and discover quickly both two and three-dimensional projections. Their reaction to the variety of variables that can be plotted seems to depend upon their educational level. The beginning student looks only at plots of position and time coordinates. The odds are against his discovering anything about energy conservation, momentum, angular momentum or motion in phase space until he has received some formal instruction. Our observations seem supportive of Bunderson's findings that directed instruction is more efficient than a pure discovery approach for the average or below average student.[9] In its present form MOTION serves these students by offering them a universe of dissimilar motions in which to test their newly learned abstractions.

Graphics in Student Programming

Our computer usage with students has involved students interacting with canned programs such as those just described, graphic and non-graphic, and it has also demanded that students write their own programs for solving physics problems. In the beginning course the two usages are about equal; in both cases comments of students at the end of the course suggest that we are at a reasonable level with regard to the amount of usage, although our students "vote" more favorably for dialogs than for computation. The total computer usage in the beginning course is about an hour and a half a student each week.

As of this writing we provide no graphic facilities which average students can use in their own programs. Knowledgable students could use our general graphic software, but this demands more knowledge than we can reasonably expect from many beginning physics students. Many students resort to character-type plotting.

Data to be graphed in physics programs is primarily array data. X and Y or X and Y and Z coordinates are calculated for many, many points, and then the resulting arrays are converted into lines

on the screen. Using graphics naturally within a student-written program depends on the ability of the programming language to conveniently construct and manipulate arrays of data. Two existing interactive graphics systems, the Culler-Fried and the Harvard TACT systems, both oriented toward easy manipulation of graphic material, are also array-oriented languages. However, these specialized languages are available in only very few places.

Of the common general purpose languages we might use with physics students, the one presenting the best array capabilities and therefore the one most suitable for graphics is APL. APL has other advantages as an introductory language for students, making it the language of choice if all current languages were available in a given location.[10]

Since APL can use array arguments to functions, several natural ways for graphics are available. Some experimentation with running systems will be useful for determining which of these is both most natural for the experienced APL user and easiest to manipulate for the beginning user. If A and B are arrays of the same length containing the data the following APL command, for example, might generate the graph:

```
DRAW    A VS B
```

Just as in dialog graphics, we need windowing and scaling, so additional functions are also necessary. And 3-D graphing should also be allowed.[11]

We hope to have a running APL graphic system soon, so we can gather experience with students. In generating teaching material we believe it essential to interact at all stages with students and to adapt the form and structure in ways that are amenable to them, rather than forcing them to match the software.

Conclusions

We have reported on graphic development and plans within the Physics Computer Development Project. Some of the ways outlined are dependent on the physics teaching material, so other areas may find other modes more natural. It may turn out that graphics are not more effective than cheaper methods of computer output in some areas. We believe that in physics the case for graphics is already strong and we believe that the potentialities for the future are great. We encourage others to experiment both within physics and in other areas to learn the capabilities of graphic teaching materials.

References

1. Bork, A., The Computer in a Responsive Learning Environment - Let a Thousand Flowers Bloom.

2. Bork, A. and Ballard, R., The Physics Computer Development Project.

3. Bork, A. and Mosmann, C., Teaching Conversations with the XDS Sigma 7 - System Description.

   Mosmann, C. and Bork, A., Teaching Conversations with the XDS Sigma 7 - System Users Manual.

   Warner, E. and Bork, A., Teaching Conversations with the XDS Sigma 7 - System Maintenance Manual.

4. Bork, A., Warner, E., and Collins, J., Teaching Conversations with the XDS Sigma 7 - Graphic Dialog Facilities.

5. Bork, A., Terminals for Education.

6. Bork, A., Inexpensive Timeshared Graphics with the Sigma 7.

7. Bork, A., Luehrmann, A. and Robson, J., Introductory Computer-Based Mechanics.

8. Taylor, E. and Wheeler, J., Spacetime Physics.

9. Bunderson, C., Instructional Software Engineering.

10. Bork, A., Science Teaching and Computer Languages.

11. Bork, A., Graphics in APL.